# Hierarchical Spatial Aggregation for Level-of-Detail Visualization of 3D Thematic Data

JAN OLE VOLLMER and MATTHIAS TRAPP, Hasso Plattner Institute, Faculty of Digital Engineering, University of Potsdam, Germany
HEIDRUN SCHUMANN, University of Rostock, Germany
JÜRGEN DÖLLNER, Hasso Plattner Institute, Faculty of Digital Engineering, University of Potsdam, Germany

Thematic maps are a common tool to visualize semantic data with a spatial reference. Combining thematic data with a geometric representation of their natural reference frame aids the viewer's ability in gaining an overview, as well as perceiving patterns with respect to location; however, as the amount of data for visualization continues to increase, problems such as information overload and visual clutter impede perception, requiring data aggregation and level-of-detail visualization techniques. While existing aggregation techniques for thematic data operate in a 2D reference frame (i.e., map), we present two aggregation techniques for 3D spatial and spatiotemporal data mapped onto virtual city models that hierarchically aggregate thematic data in real time during rendering to support on-the-fly and on-demand level-of-detail generation. An object-based technique performs aggregation based on scene-specific objects and their hierarchy to facilitate per-object analysis, while the scene-based technique aggregates data solely based on spatial locations, thus supporting visual analysis of data with arbitrary reference geometry. Both techniques can apply different aggregation functions (mean, minimum, and maximum) for ordinal, interval, and ratio-scaled data and can be easily extended with additional functions. Our implementation utilizes the programmable graphics pipeline and requires suitably encoded data, i.e., textures or vertex attributes. We demonstrate the application of both techniques using real-world datasets, including solar potential analyses and the propagation of pressure waves in a virtual city model.

CCS Concepts: • **Human-centered computing → Geographic visualization**; *Visualization techniques*;

Additional Key Words and Phrases: Level-of-detail visualization, spatial aggregation, real-time rendering

**ACM Reference format:**
Jan Ole Vollmer, Matthias Trapp, Heidrun Schumann, and Jürgen Döllner. 2018. Hierarchical Spatial Aggregation for Level-of-Detail Visualization of 3D Thematic Data. *ACM Trans. Spatial Algorithms Syst.* 4, 3, Article 9 (September 2018), 23 pages.
https://doi.org/10.1145/3234506

Authors' addresses: J. O. Vollmer, M. Trapp, and J. Döllner, Research Group Computer Graphics Systems, Hasso Plattner Institute, Faculty of Digital Engineering, University of Potsdam, Prof.-Dr.-Helmert-Straße 2-3, 14482, Potsdam, Germany; emails: {jan.vollmer, matthias.trapp, juergen.doellner}@hpi.de; H. Schumann, Computer Graphics, Institute for Computer Science, University of Rostock, Albert-Einstein-Straße 22, 18059, Rostock, Germany; email: heidrun.schumann@uni-rostock.de.

## 1 INTRODUCTION

In cartography and geo-information science, *thematic maps* are often used to visualize semantic data (a *theme*) with a spatial reference, i.e., where each of the thematic data points is associated with a geographic location. For the purpose of our work, a thematic map combines two major components: (1) a *reference geometry* with a well-defined shape and (2) the abstract *thematic data* without inherent shape. Most common is the visualization of 2D thematic data with an underlying map as reference geometry, whereas in this article, we focus on the visualization of 3D spatial and spatiotemporal thematic data mapped onto virtual 3D city models, though our techniques could theoretically be applied to arbitrary models. They assume the thematic data to be suitably encoded for real-time rendering based on the programmable graphics pipeline, namely, as 2D surface or 3D volumetric texture, or alternatively as per-vertex, per-primitive, or per-object attributes. Mapping the thematic data onto the reference geometry as its natural reference frame facilitates the goal of understanding the data and perceiving emerging patterns with respect to its spatial reference. This type of visualization was used long before the invention of computers, an early example being the map of the Golden Square, London, published by the physician Jon Snow in 1855 displaying the number of cholera victims per house as stacked bars, leading the creator to the source of the outbreak [44].

Today, the goal of thematic data visualization has not changed, but the amount of data that is captured and available has increased, requiring the support of complex, computer-based visualization systems and metaphors that can process and display both abstract and geometric data. The increased amount of data leads to an increasing information density in the resulting visualization, causing visual clutter that impedes perception. To reduce the amount of visual clutter in 2D maps, a variety of cartographic *generalization* techniques have been presented that omit unnecessary details by replacing cartographic elements with representations of higher abstraction and smaller scale [23]. In this context, generalization techniques for thematic data are typically based on the principle of *aggregation*, i.e., collapsing multiple data points with respect to the value or spatial domain by means of an *aggregation function*, such as mean or maximum. A number of researchers have proposed aggregation techniques for 2D thematic data [16, 18].

In contrast, to the authors' knowledge, there are no approaches for the aggregation of 3D thematic data, i.e., attributes associated with primitives of a possibly animated 3D geometrical scene. Typical examples are *geovirtual environments (GeoVEs)*, such as 3D virtual city and landscape models, which are important tools to communicate geospatial information [50], e.g., solar potential [17], visibility information [11], and pressure data [33]. Thematic data represents one of the most prominent data types visualized within GeoVEs, as *thematic data analysis*, i.e., the identification, analysis, and reporting of patterns within this data, is a major task in geographic information system (GISs).

### 1.1 Level of Detail in Visualization

By varying the degree of aggregation (i.e., the number of data points collapsed to a single value), multiple *levels of detail (LoDs)* can be generated and selectively applied during rendering by means of interactive LoD visualization techniques to reduce information density on screen, thus supporting exploration and facilitating analysis as well as reasoning [14]. 3D visualization often exhibits higher degrees of information overload and visual clutter compared to 2D visualization; hence, particularly in this case, LoD generation and visualization techniques are required. In general, Thöny et al. [47] predict a trend toward more interactive terrain visualizations within GISs that, coupled with growing datasets, requires sophisticated LoD and multiresolution rendering techniques.

To efficiently handle increasing data complexity with respect to effective visual communication, the process of visualization can apply different LoD approaches at various stages. In general, one can distinguish between techniques for the generation of LoDs, as well as techniques for the selection and combination of appropriate LoDs during rendering. The former type can be further categorized into the following:

**Geometric Level of Detail:** This type is well established in the computer graphics community [34]. There are a number of specific approaches for 3D GeoVEs established, including geometric *levels of abstraction (LoAs)* [41, 51] and specific approaches for buildings [12]; however, these techniques do not consider associated thematic attribute data.

**Data Level of Detail:** Approaches handling the aggregation of data are well researched in visualization [16, 43]; however, most of these techniques operate in the data domain and do not consider the spatial reference frame of thematic data, or are limited to a 2D reference frame.

The latter can comprise *overview+detail* or *focus+context* techniques [7, 49] to support Shneiderman's [42] *Visual Information Seeking Mantra*: "Overview first, zoom and filter, then details-on-demand." Both categories combine multiple levels of detail, typically a coarse level used for an overview and one or more fine levels for specific *regions of interest (RoIs)*. While overview+detail techniques display these separately, e.g., using two views, focus+context visualizations combine multiple detail levels in a single view based on user interactions. In general, different abstractions of information are taken into account, but abstract and geometric data are typically not specifically distinguished.

## 1.2 Challenges and Contribution

While the generation of geometric levels of detail or levels of abstraction, as well as LoD, in viewing is already covered by various previous research, we focus on computing and visualizing levels of detail for thematic data in 3D GeoVEs. Our LoD aggregation and visualization techniques were developed with the following challenges in mind:

- Pregeneration of LoDs is infeasible for spatiotemporal thematic data due to storage requirements, and hence an on-demand generation in real-time is necessary.
- As thematic data can be encoded in different ways, aggregation must operate independently of a specific representation.
- The data structures to store LoDs must be compact enough to include high detail levels, while fitting into GPU memory.
- LoD sampling must be efficient enough to be incorporated into a real-time rendering process.
- LoD selection must be dynamic and controllable through existing LoD interaction techniques.

With respect to the challenges above, we present the following approach: after applying Crassin and Green's [9] real-time voxelization algorithm to obtain an initial discretization of the scene, subsequent thematic data aggregation is performed using one of two hierarchical spatial aggregation techniques:

**Scene-based** aggregation constructs a sparse voxel octree where each node aggregates the thematic values of its eight child nodes, thus enabling LoD generation for a variety of thematic data representations without requiring any knowledge of their semantics.

**Object-based** aggregation takes into account the hierarchy of objects in the scene and computes an aggregated value for a scene object based on the thematic values of its child objects. This requires adequate encoding of objects and their hierarchies in the reference geometry representation, e.g., in the form of a scene graph where each node constitutes an object.

Further, we describe efficient data structures and sampling algorithms for both aggregation techniques, as well as their application during rendering in combination with different LoD interaction techniques, such as focus+context and detail+overview approaches to control LoD selection.

## 2 RELATED WORK

This section presents previous work in the field of level-of-detail visualization, as well as on algorithms and data structures to create and represent spatial detail levels.

### 2.1 Level-of-Detail Visualization

The presented work mainly relates to previous work in the fields of geometric LoD, spatial data aggregation, and LoD viewing using focus+context or detail+overview approaches for 3D GeoVEs.

*2.1.1 Geometric Level of Detail and Level of Abstraction.* Rendering techniques for 3D geometric LoD have been well researched during the past decades [34]. With respect to 3D GeoVEs, the LoD notion was extended to level of abstraction [41], consisting of precomputed generalization approaches at various granularity levels: from cell-based generalization variants for virtual 3D city models to the generalization of single buildings [19]. However, these approaches suffer from massive data volumes resulting from precomputations, which have to be managed during runtime and do not consider the aggregation of data associated with the 3D geometry. Our approach can incorporate cell-based LoA as reference geometry. Semmo et al. [41] present a hardware-accelerated approach for transitions between such LoAs that partially computes intermediate levels on demand. This approach significantly reduces memory consumption and can be used for different feature types within 3D GeoVEs, but does not consider thematic data attributes either.

Generic *mesh simplification* techniques can be applied in geovisualization as well to geometrically simplify models and create geometric LoDs; however, they generally exhibit one or more of the following limitations. With very few exceptions [39], most techniques operate exclusively on meshes composed of triangles [8, 10, 20, 24, 37], and some require special input data structures (e.g., half-edge structures) to increase performance [20]. Our approach does not require specific geometry representations, as long as they are suitable for rasterization with the programmable graphics pipeline. This includes other geometry types as well, e.g., point clouds. Further, only a subset of mesh simplification techniques can handle *appearance attributes* [8, 24], and thus are applicable for thematic data, while others only deal with geometry [10, 20, 37, 39]. Lastly, there are approaches utilizing the graphics processing unit (GPU) to simplify small to medium-sized meshes in real time [10, 37]; however, most techniques do not [8, 20, 24, 39].

*2.1.2 Level-of-Detail Visualization Techniques.* There are various approaches for LoD visualizations in the 3D virtual environment (VE), e.g., focusing on molecular visualization [21, 38] or graph drawings to be rendered interactively with adjustable LoD based on density fields that are computed at runtime [55]. Our approach also uses a data structure for aggregation that can be computed at runtime based on the current viewing settings. With respect to 3D GeoVEs, Zhang et al. [53] propose a conceptual model to handle semantic LoD visualization in urban environments but do not account for specifics of thematic data.

By focusing on 3D maps, Engin et al. [18] present an interactive visualization technique for thematic data based on combining precomputed detail levels facilitating faster understanding while avoiding confusion when viewing 3D maps; however, their technique is limited to 3D maps, i.e., 2D maps with an additional height component, and does not support generic 3D data. Using lens-based focus+context visualization for combining different levels of detail (e.g., geometrical and stylization variants) in 3D GeoVEs is described by Trapp [49]; however, no algorithms for the generation of thematic data detail levels are presented.

*2.1.3 Level of Detail for Thematic Data.* While spatial data aggregation is a prominent topic in the overlap between GIS and databases [26], the state of the art is focusing on 2D region-based aggregation, which alone is not suitable for the specifics of true 3D data. Considering aerial and facade texture data, Buchholz and Döllner [5] present an approach that aggregates facade texture data in multiresolution texture atlases using a scale-space method; however, it is limited to facade textures only and does not consider thematic data aggregation.

Zhang et al. [54] present an approach that utilizes multicore central processing unit (CPU) and many-core GPU hardware acceleration to support the aggregation of spatiotemporal trajectory data. Unfortunately, this approach is not able to deal with different representations of potentially dynamic thematic data. An approach for spatiotemporal aggregation of massive movement data for visual analysis is presented by Andrienko and Andrienko [1]; however, the presented aggregation technique relies on precomputation and uses a 2D map for visual communication only. Elmqvist and Fekete [16] present a model for hierarchical aggregation in information visualization for the purpose of improving the overview and scalability of large-scale visualizations.

## 2.2 Efficient Data Structures for Massive Data Aggregation

The scene-based aggregation technique presented in this article uses a compact volumetric data structure supporting fast creation and access from the GPU that is based on existing work described below. Besides attributed polygons embedded in 3D space, the *voxel* primitive, i.e., a discretization of three-dimensional space, can be used to effectively represent 3D thematic data. *Voxelization* describes the process of generating a *voxel space* based on 3D polygonal input models [13, 15], allowing multiple detail levels through different resolutions of the voxel space. A set of detail levels with suitable resolutions can be combined to form an *octree* [2], i.e., a hierarchical data structure for efficient traversal of the voxelspace that can be used for visualization-based analysis [4]. A *sparse voxel octree (SVO)* is a compact representation of an octree [30]. To conserve memory, a *sparse* octree only stores nodes actually covered by the source geometry [2].

Crassin and Green [9] present a technique for generating an SVO representation of a 3D polygonal input scene in real time using the rasterization pipeline of modern graphics hardware. We adapt this technique for thematic data aggregation and add efficient sampling methods for geometrically complex 3D GeoVEs in terms of runtime complexity. However, a major limitation of the voxel representation approach is the induced memory consumption. Kämpe et al. [27] present an approach for managing high-resolution SVOs as directed acyclic graphs (DAGs), which can counterbalance this limitation. Although it is not the focus, our approach can be extended to support DAGs. Labschütz et al. [29] present a hybrid data structure for sparse volumetric data that can compose sparse octree, as well as dense volume array parts, while retaining fast access from GPUs; however, the optimization of this structure requires additional processing steps that fail to meet the real-time requirement.

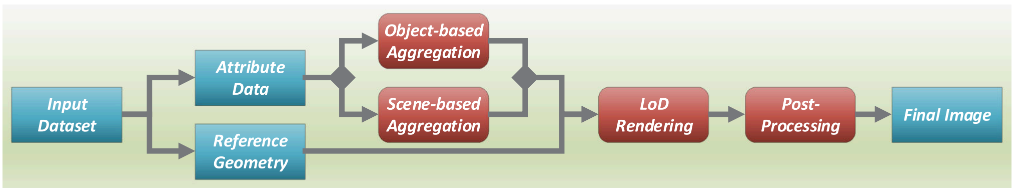


Fig. 1. Conceptual overview of the presented level-of-detail technique for thematic data visualization.

## 3 CONCEPTUAL OVERVIEW

This section covers fundamental concepts and a general process overview for the presented thematic data aggregation techniques.

### 3.1 Conceptual Outline

The system we designed and implemented provides two real-time aggregation techniques for thematic data, as outlined in Figure 1: (1) an object-based technique that aggregates thematic data based on a set of objects and their hierarchy as defined in the input data, thus supporting per-object analysis, and (2) a scene-based technique that aggregates data based on spatial location only, thus enabling LoD generation for a variety of thematic data representations without requiring any knowledge of their semantics. In this article, we focus on generating LoDs for the thematic data and do not consider geometric LoDs for the reference geometry; however, the resulting thematic LoD representations can theoretically be mapped onto reference geometries other than the original, provided these have the same spatial reference frame and object hierarchy, or a subset thereof. This includes geometric LoDs created by external tools.

During rendering, the level of detail is determined on a per-pixel basis using a number of different LoD control mechanisms, including focus+context lenses, distance-to-camera estimation, and manual control. The results of either aggregation technique are then mapped onto the original reference geometry at the determined LoD and with a suitable color mapping scheme based on the work of Harrower and Brewer [22] and Engel et al. [17]. Optionally, value quantization can be applied during color mapping with the number of intervals depending on the LoD as well. A postprocessing stage applies visual enhancements according to Engel et al. [17].

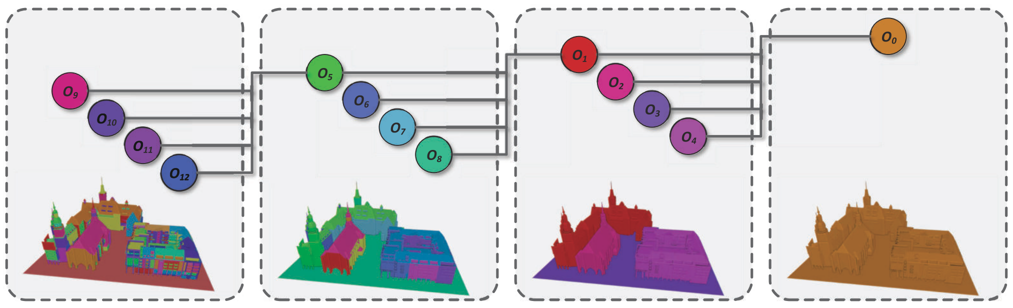
### 3.2 Preliminaries and Assumptions

To present aggregated thematic data for specific parts of the reference geometry (e.g., walls, buildings, blocks, districts), the input scene must identify objects at these different granularity levels, thus forming an *object hierarchy* as shown in Figure 2. A scene graph is a typical example of an object hierarchy, where each node constitutes an object.

Since the presented aggregation techniques rely on fast, hardware-accelerated discretization of three-dimensional space using standard hardware rasterization, the attribute data must be accessible as a normalized scalar value ($v \in [0, 1] \subset \mathbb{R}$) at the per-fragment level. The encoded attribute values may be associated with its geometrical context in different ways, which can be categorized into (1) *geometric attributes*, such as per-object, per-primitive, and per-vertex data, and (2) *raster data*, such as 2D surface and 3D volume textures.

### 3.3 Process Overview

Figure 3 shows a process overview for the presented techniques, comprising three major stages: (1) *data preprocessing* (Section 3.3.1), (2) *level-of-detail generation* (Section 3.3.2), and (3) *scene*



(a) Level 3: Elements. (b) Level 2: Walls. (c) Level 1: Buildings. (d) Level 0: Blocks.

Fig. 2. Exemplary object hierarchy of a virtual city model (Chemnitz) with four levels. False coloring is used to delineate individual objects.

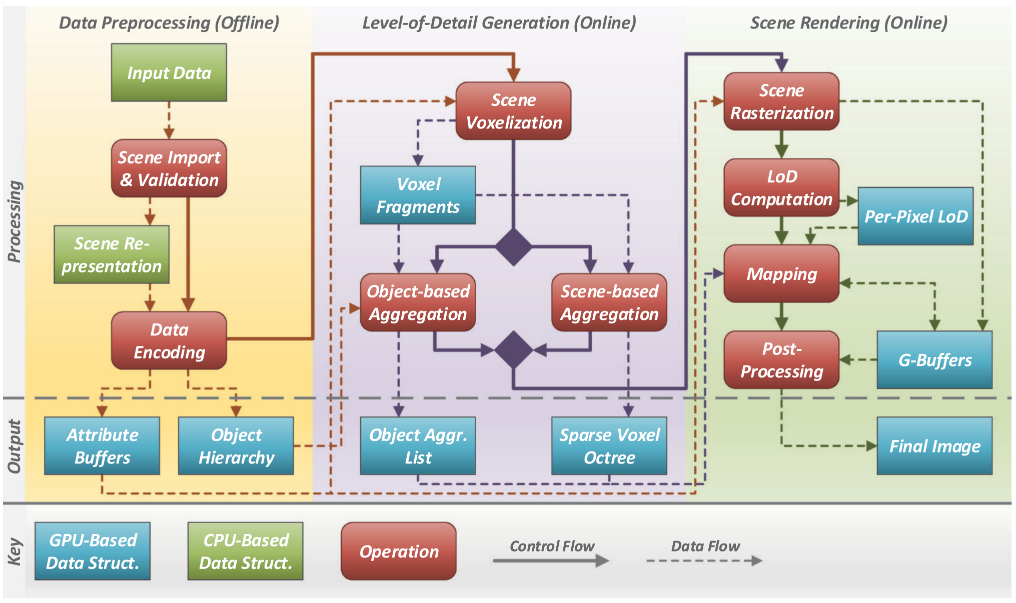


Fig. 3. Schematic overview of the presented system with operations and data structures required to realize the concept illustrated in Figure 1.

*rendering* (Section 3.3.3). While the last two stages are executed *online*, i.e., in real time, the first stage is designed as an *offline* stage, i.e., executed once per dataset without time constraints. This is required to transform the input data into a representation suitable for real-time LoD generation. The design and implementation of this process is based on the *OpenGL 4.5* graphics application programming interface (API) [40] in combination with the *OpenGL Shading Language (GLSL) 4.50* [46].

*3.3.1 Data Preprocessing.* The data preprocessing stage is responsible for reading the input dataset and creating a GPU-based representation suitable for real-time LoD generation and rendering. This includes (1) the decoding and validation of various input formats (e.g., Wavefront OBJ, or 3DS) and subsequently (2) encoding into an internal scene representation consisting of a set of

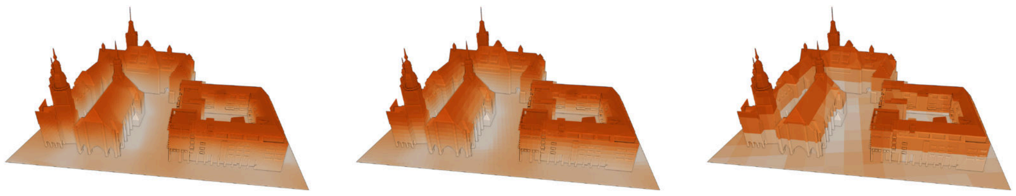


Fig. 4. Scene-based aggregation of surface solar potential in a virtual city model (Chemnitz) at decreasing detail levels (left to right).

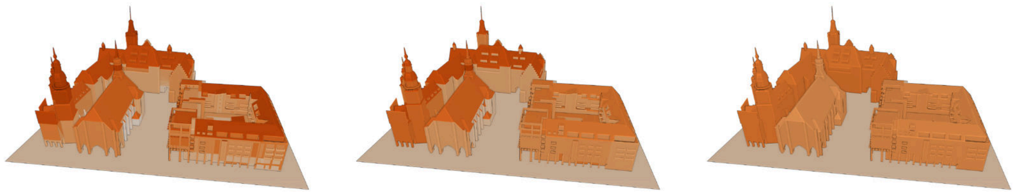


Fig. 5. Object-based aggregation of surface solar potential in a virtual city model (Chemnitz) at decreasing detail levels corresponding to the object hierarchy levels presented in Figure 2 (left to right).

binary attribute buffers suitable for GPU-based operations. In addition, the preprocessing stage (3) generates object ID attributes during import, as well as (4) a GPU-based representation of the scene's object hierarchy. As these steps are only required once per dataset, the results of this stage can be persisted on disk and reused during later executions to speed up loading time.

*3.3.2 Level-of-Detail Generation.* The level-of-detail generation stage performs thematic data aggregation using the scene-based or object-based aggregation technique. Both techniques are capable of operating in real time to support the on-the-fly generation of LoDs required for the visualization of dynamic thematic data. Prior to aggregation, the input scene is discretized into a set of voxel fragments using a process based on the first stage of Crassin and Green's [9] hardware-accelerated SVO construction technique adapted to the requirements of thematic data aggregation; i.e., each voxel fragment stores a thematic attribute value and object ID as associated with the reference geometry at the fragment's position. Section 4 describes the aggregation techniques and their respective data structures in detail.

*Scene-Based Aggregation.* Subsequently, the scene-based aggregation technique constructs a sparse voxel octree, thus creating multiple detail levels by aggregating attribute values based on voxel positions (Figure 4). This process is based on the second stage of Crassin and Green's [9] technique, extended to support multiple aggregation functions, such as minimum, maximum, or average, thus facilitating thematic data analysis. As the aggregation is solely based on the position in three-dimensional space, no additional sampling coordinates are required and the existing position vertex attribute suffices. Section 4.2.1 provides a detailed description of this aggregation technique.

*Object-Based Aggregation.* The object-based aggregation technique achieves multiple detail levels by aggregating the voxel fragments' attribute values based on the object IDs and their hierarchy (Figure 5) and storing the results in a simple linear list indexed by the object ID; thus, no additional sampling coordinates are required either. The object-based aggregation technique supports the same aggregation functions as the scene-based aggregation technique described above. Section 4.2.2 describes this aggregation technique in detail.

*3.3.3 Scene Rendering.* The scene rendering stage renders the reference geometry and maps the aggregated thematic data onto it. It uses a *deferred shading* [45] approach that rasterizes the reference geometry into a set of G-Buffers storing various geometric attributes and defers sampling of aggregation data structures, as well as lighting operations to an image-based shading pass. This reduces the number of sampling operations to the number of fragments actually visible in the output image, at the cost of increased memory consumption due to the G-Buffer representations.

A number of interactive LoD mechanisms (introduced in Section 3.1) can be configured and applied to compute a normalized LoD value $l \in [0, 1] \subset \mathbb{R}$ on a per-pixel basis. During the subsequent mapping step, the LoD value is passed to the sampling routines of either aggregation technique to determine the detail level to sample, resulting in a normalized thematic attribute value. Optionally, LoD value quantization can be applied to this attribute value before the application of a color mapping. The resulting color is shaded using the Blinn-Phong lighting model [3] and finally mapped onto the reference geometry. An additional postprocessing pass applies visual enhancements, such as ambient occlusion, or edge enhancement. Section 5 describes the rendering and mapping process in detail, as well as the LoD interaction techniques.

## 4 REAL-TIME LEVEL-OF-DETAIL GENERATION

This section covers the generation of data structures representing discrete spatiotemporal thematic data, as well as subsequent thematic data aggregation and level-of-detail generation. Initially, attributed geometric data is voxelized, which then serves as the basis for two spatial aggregation techniques: (1) scene-based aggregation (Section 4.2.1) and (2) object-based aggregation (Section 4.2.2).

### 4.1 Encoding of Object Identifier and Hierarchy

Prior to the real-time LoD generation stage, object hierarchy information contained in the input data must be converted to a GPU-suitable representation to support object-based thematic data aggregation. This implies that the reference geometry must identify objects at different granularity levels, as well as their hierarchy. The meaning of the term "object" depends on the application; typically, this refers to a unique, self-contained entity of the input scene. Examples from the context of the virtual city model are cities, districts, blocks, buildings, walls, or construction elements, as depicted in Figure 2. Another example is the scene graph data structure, where each node constitutes an object and each level constitutes a corresponding level in the object hierarchy.

*Unique Object Identifier.* To distinguish objects during rendering, they must be uniquely identified using an integral numerical identifier denoted as *object ID*. During file import, the object ID is generated by assigning consecutive IDs bottom-up, i.e., starting at the leaf objects with ID 0 up to the root node receiving the highest ID value (Figure 6). For leaf objects, the IDs are stored as an additional vertex attribute, while IDs of objects on higher levels are encoded in the object-hierarchy matrix described below. The object ID attribute is encoded as an unsigned integer with 8 bit, 16 bit, or 32 bit (depending on the number of objects), thus enabling its direct use as a list index during aggregation.

*Object Hierarchy Matrix.* Further, the object hierarchy of the reference geometry must be encoded into a data structure that is suitable for fast access from shader programs during rendering and thematic data aggregation. Instead of encoding the hierarchy explicitly using a tree, it is implicitly encoded based on a two-dimensional matrix denoted as *object-hierarchy matrix* (Figure 6). This has two major advantages: (1) it can be effectively stored in video memory using a single texture, and (2) it enables efficient access at shader runtime with only a single texture sampling

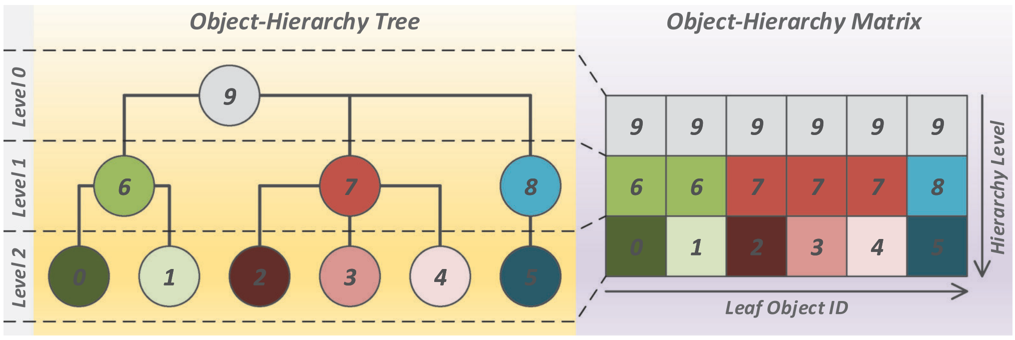


Fig. 6. Encoding of a given object-hierarchy tree (left) into an object-hierarchy matrix (right) for efficient access from shader programs. Colors and numbers represent different objects and their IDs.

operation required. The increased memory consumption compared to trees due to the repeated storage of object IDs on higher levels is negligible.

The two matrix dimensions represent the leaf object ID and hierarchy level. Each leaf object ID is inserted into the matrix at the lowest level, and corresponding cells on higher levels are filled by traversing the hierarchy tree bottom-up for each leaf object. During rendering, leaf object IDs are accessible from shader programs as a vertex attribute and can be used as sampling coordinates to retrieve corresponding object IDs on higher levels using a single texture fetch operation.

## 4.2 Spatial Data Aggregation Techniques

Both aggregation techniques are based on the discretization of three-dimensional space contained in the input scene's minimal bounding cube into *voxels*. To perform the voxelization process in real time during visualization, we adapted the hardware-accelerated technique presented by Crassin and Green [9]. This technique rasterizes the input scene's primitives at the resolution of the voxel space using the hardware rasterizer of modern GPUs to create a set of *voxel fragments*. Each voxel fragment consists of a position in voxel space coordinates, the thematic attribute value associated with the reference geometry at this position, and the corresponding object ID. As the original primitives may overlap, more than one resulting voxel fragment may occupy a single voxel, hence the term voxel fragment.

Based on results of the scene voxelization, the now spatially discrete thematic attribute values are aggregated yielding different detail levels. We propose two *hierarchical aggregation types*: (1) *scene-based aggregation* that transfers the voxel fragment list into a sparse voxel octree inherently containing different tree and therefore detail levels, and (2) *object-based aggregation* that aggregates voxels according the specific object hierarchy represented by the reference geometry.

*4.2.1 Scene-Based Data Aggregation.* To perform hierarchical spatial aggregation independent from the reference geometry, a *sparse voxel octree* [30] is used as a hierarchical spatial data structure. The root node of an SVO on level 0 consists of a single voxel covering the entire bounding cube, i.e., its size in voxel-space coordinates is $2^{n-1}$ in each dimension. Further levels are created by subdividing each node on level $k$ into $2^3 = 8$ child nodes on level $k + 1$, and thus the voxel-space size of a node on level $k$ is $2^{n-k-1}$ in each dimension. The subdivision process ends when the number of nodes on the last level corresponds to the voxel-space resolution used during scene voxelization; i.e., the leaf node's positions and sizes correspond to the previously created voxel fragments. To conserve memory, only nodes covering a part of the reference geometry are stored, while empty nodes are discarded.

Such an octree data structure is used for spatial aggregation by associating each node with a thematic data value, and thus each octree level represents a thematic detail level in which each node aggregates the values of its eight child nodes. The presented GPU-based SVO construction algorithm builds on the algorithm described by Crassin and Green [9]. The construction process consists of three steps suitable for GPU-based execution using compute shaders:

**Octree Generation.** The spatial octree structure is created using a top-down approach starting at the root node and using the list of voxel fragments created in the scene voxelization stage by repeating the following steps for each new level to create: (1) for each voxel fragment, traverse the existing tree down to the node on the current level $k$ that corresponds to the fragment's position and mark this node; (2) for each node marked in the previous step, create eight empty child nodes on level $k + 1$ by allocating their storage space.

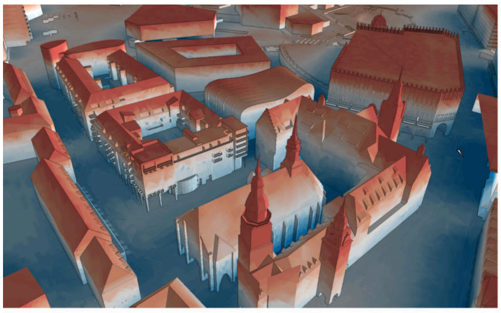
**Leaf-Node Initialization.** After constructing the SVO structure, the tree's leaf nodes on the last level are initialized with the thematic attribute values associated with the voxel fragments. Since more than one voxel fragment can correspond to a node, their values must be merged using a similar approach to the value aggregation described in Section 4.2.3.

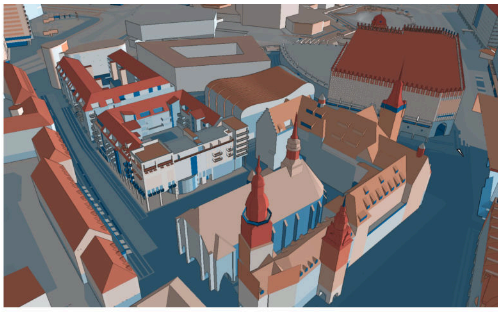
**Value Aggregation.** Thematic attribute values for nodes on upper levels are computed using a bottom-up approach in which the attribute value of a node on level $k$ is determined by the aggregation functions described in Section 4.2.3 based on the values of its (up to) eight child nodes on level $k + 1$, until reaching the root node.

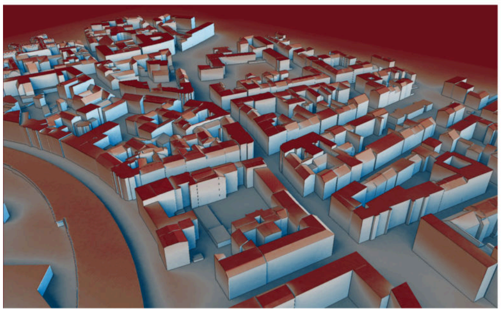
*4.2.2 Object-Based Data Aggregation.* In addition to scene-based aggregation, object-based data aggregation can be computed based on the object hierarchy (parts-whole relationship) given by the structure of the reference geometry. For buildings of a 3D virtual city model, hierarchy information can be provided by various data exchange formats such as CityGML [28] or IFC [32] for building information models. With respect to this, Figure 7 shows a comparison between nonaggregated thematic data (Figure 7(a) and Figure 7(c)) and a single LoD with per-object aggregation (Figure 7(b) and Figure 7(d)). Here, original 3D thematic data that spatially contributes to specific geometry objects (e.g., building roofs or walls) is aggregated with respect to the object-hierarchy matrix. LoD visualization using object-based aggregation can facilitate the visual analysis of data constituting visual clarity in the presentation of the aggregates by supporting the perception of individual structures in the 3D scene.

As the object-hierarchy matrix already encodes a hierarchy structure (and therefore determines the detail levels) for the object-based aggregation, there is no need to create a specialized data structure for aggregated thematic attribute values. Instead, a simple list indexed by the object identifier suffices. The object-based aggregation process consists of two steps performed for each voxel fragment created during scene voxelization: (1) copy and merge the attribute value associated with the fragment into the aggregation list at the leaf object ID associated with the fragment using the aggregation functions described in Section 4.2.3, and (2) look up the identifiers of parent objects in the object-hierarchy matrix and repeat step (1) for each of them.

This approach is not limited to object hierarchy data only. Given a representation of specific *aggregation regions* (e.g., represented using 2D shape files or raster maps), values can be aggregated for each region using the same algorithm as described before. Instead of using the object identifier, the spatial position of each voxel is used for its association with the respective regions. Figure 8 shows an example for this type of object aggregation performed for the different districts of three virtual 3D city models. One can easily perceive differences of the solar potential between the respective districts, e.g., large differences in Boston (Figure 8(a)) or almost no differences in Berlin (Figure 8(c)).


(a) Original 3D thematic data.


(b) Per-object aggregation.


(c) Original 3D thematic data.

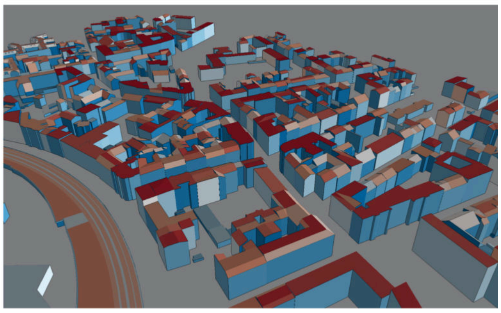
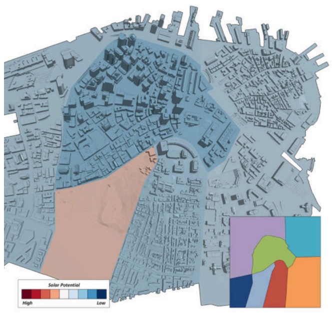
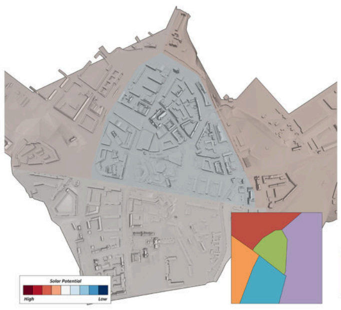

(d) Per-object aggregation.

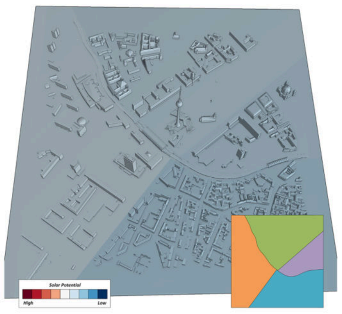
Fig. 7. Comparison between the visualization of original 3D thematic data and the results of the object-based aggregation at the highest level of detail (datasets: Chemnitz (top), Berlin (bottom)).



(a) Boston.



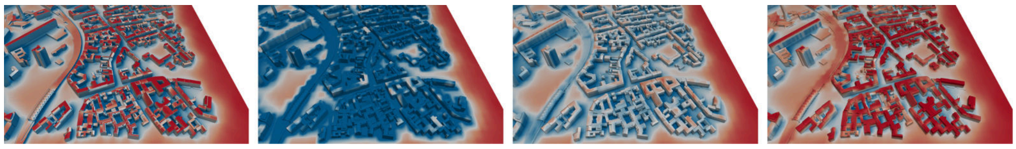
(b) Chemnitz.



(c) Berlin.

Fig. 8. Examples of object-based aggregation using specific aggregation regions corresponding to districts.

*4.2.3 Aggregation Functions.* To comprehensively support visual analysis, both aggregation techniques provide different aggregation functions, each operating on two inputs. During aggregation, the system simultaneously computes four functions using atomic operations suitable for execution within a compute shader: (1) *minimum*, (2) *maximum*, (3) *sum* (computes the total sum of all attribute values contributing to this aggregate), and (4) *count* (computes the total number of attribute values contributing to this aggregate). The additional average function can be computed from the final sum and count. Figure 9 compares exemplary results of scene-based aggregations



(a) Original data. (b) Minimum. (c) Average. (d) Maximum.

Fig. 9. Comparison of visualization results using different aggregation functions applied during scene-based aggregation of solar potential data for a virtual city model (Berlin).

using minimum, maximum, and average functions. For example, using a minimum aggregation function emphasizes the location of regions with lower solar potential and thus visually represents a conservative assessment, while using a maximum aggregation function would communicate an overoptimistic assessment of a spatial region.

## 4.3 Implementation Details

The following sections provide technical insights into key aspects of the techniques described in this chapter.

*4.3.1 Direct versus Indirect Dispatch of Compute Shader Programs.* Several of the steps described above are implemented using compute shaders executed with one invocation per voxel fragment. The scene voxelization process stores generated voxel fragments in a linear list of a predefined size that was previously allocated from video memory by means of a buffer object. To allocate storage space for individual voxel fragments within this list, an *atomic counter* [40, p. 139–141] object is used: for each new fragment, the counter is incremented and the result used as a list index. After all fragments are written, the counter contains the number of voxel fragments; i.e., its value determines the number of computer shader invocations for subsequent steps. The counter value is stored in a separate buffer object in video memory to facilitate fast accesses from shader programs.

The straightforward approach to dispatch a compute shader based on this number would be to transfer the content of the atomic counter buffer back to main memory and then call `DispatchCompute` [40, p. 521] with a corresponding number of compute shader workgroups. Using this approach can lead to a major performance degradation, as the transfer of buffer content constitutes an *implicit synchronization* point between the application and the OpenGL driver, causing a pipeline stall [25]. Using asynchronous transfers is not an option, as the data is required immediately to dispatch the next compute pass.

Instead, the more efficient solution is to use the `DispatchComputeIndirect` command [40, p. 522] that reads workgroup specifications from a buffer object, and thus no data transfer to application memory is required. To create the buffer-based specification, an extra compute shader program is required, incorporating the device-specific workgroup count limits.

*4.3.2 Atomic Operations on Floating-Point Values in GLSL.* For both aggregation techniques, the thematic data values associated with the generated voxel fragments are initially merged by launching a compute shader program per fragment. Each invocation updates the target aggregates, either per object or per leaf voxel, stored in a global aggregation buffer, thus performing read-modify-write operations that potentially conflict with other shader invocations accessing the same target. To synchronize accesses, atomic operations are required. While the GLSL specification provides atomic functions, these are only defined for integer values [46, p. 175–176]; however, the thematic attribute values to be aggregated are scalar floating-point values. The *NV_shader_atomic_float*[1]

[1]https://www.opengl.org/registry/specs/NV/shader_atomic_float.txt.

```
// Buffer storing aggregated values, index by the object or node ID
layout(std430, binding = 0) buffer AggregationBuffer {
    float minimum[];
};
void main() {
    // Get thematic value from voxel fragment
    const float value = ...;
    // Determine buffer index
    const uint index = ...;
    // Swap values until the old value is not smaller
    // than the previous value, i.e., _not_ the minimum
    float newMin = 1.0;
    float currentMin = value;
    while (currentMin < newMin) {
        newMin = currentMin;
        currentMin = atomicExchange(minimum[index], newMin);
    }
}
```

Listing 1. GLSL code of a compute shader demonstrating minimum aggregation based on `atomicExchange`.

extension to the OpenGL API provides two atomic functions for floating-points values: `atomicAdd` and `atomicExchange`, allowing a straightforward implementation of the sum and count aggregation using `atomicAdd`.

Atomic functions for the computation of minimum and maximum are still missing, and hence these must be simulated, as demonstrated in Listing 1: after determining the thematic value of the current voxel fragment, as well as the target index of the object or leaf node, the minimum of the current aggregate and the fragment's value is computed by executing `atomicExchange` in a loop on the aggregate and the previous iteration's result until the returned value (i.e., the old aggregate value) is greater than or equal to the value returned in the previous iteration. As this cannot be the minimum value, the loop can terminate at this point. In case multiple shader invocations access the same target using this algorithm, each invocation loops until holding on to a value that cannot be the minimum, and thus the value remaining in the global buffer after the last invocation finished must be the minimum. The maximum aggregation function is implemented using the same approach.

## 5 REAL-TIME LEVEL-OF-DETAIL VISUALIZATION

This section covers techniques for the interactive level-of-detail visualization of thematic data in 3D GeoVE, comprising two steps: (1) interactive LoD control using different techniques (Section 5.1) and (2) subsequent LoD mapping that samples the results of scene-based and object-based aggregation accordingly to associate thematic detail levels with the three-dimensional reference geometry (Section 5.2). The provided options allow for comprehensive visual analysis of thematic data in the 3D geovirtual environment. The LoD visualization process uses a *deferred shading* approach that performs LoD computation and sampling on an image-based representation of the reference geometry using an image-based shading pass denoted as *mapping pass*. See Section 5.2 for details.

### 5.1 Level-of-Detail Control

This section describes techniques to interactively control the mapping of thematic detail levels onto the reference geometry. Each of these techniques is evaluated on a per-pixel basis and results in a normalized LoD value $l \in [0,1] \subset \mathbb{R}$, whereby $l = 0$ refers to the most generalized (*lowest*) detail level, while $l = 1$ refers to the most detailed (*highest*) level. Prior to its application during sampling, $l$ is scaled to a user-configurable range $[l_{min}, l_{max}] \subset \mathbb{R}$ to limit the set of available detail levels.

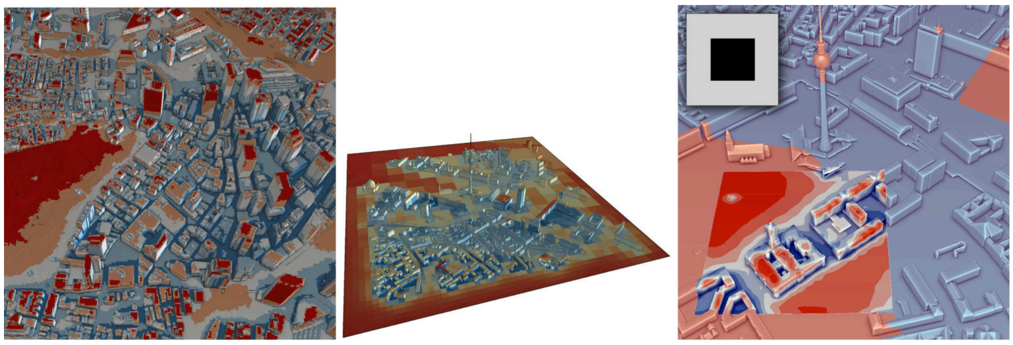

(a) Global LoD. (b) Distance-to-camera. (c) Lens-based focus+context visualization using a scene-lens.

Fig. 10. Exemplary results using different interaction techniques to control the level of detail (dataset: Berlin).

*Global Level of Detail.* A straightforward application is the usage of a single *global* LoD value for the complete scene (Figure 10(a)). This can be used for visual analysis of thematic data by gaining an overview, e.g., find regions of high or low solar potential by subsequently decreasing the level of detail of scene-based aggregations manually. It further can be used to explore the contributions of solar potentials to individual parts of particular buildings using object-based aggregation.

*Distance to Camera.* This a typical way of using level of detail: LoD values are adjusted according to the distance in camera space (Figure 10(b)). Automatic adjustment of LoD values to zoom levels facilitates free navigation and exploration within 3D GeoVE. It usually exhibits high LoD near the virtual camera or at close zoom stages and low details in the distance or at high zoom stages [18]. The LoD value $l$ is computed using the linearized fragment depth $f_z \in [0, 1] \subset \mathbb{R}$ [31] and a transfer function $t_d$ for mapping between fragment depth and LoD value: $l := t_d(f_z)$. The transfer function $t_d$ is encoded as a user-configurable, one-dimensional grayscale texture and thus can have linear or nonlinear characteristics.

*Lens-Based Focus+Context Visualization.* To interactively control the thematic LoD values displayed at spatial regions within a visualization and to further facilitate viewing level of detail, lens-based metaphors are supported [48]. They enable a user to control the position and size of multiple focus *regions of interest* within an aggregated context (Figure 10(c)). For the purpose of this work, lenses are basically—but not limited to—two-dimensional RoI masks represented by 2D textures containing grayscale values that encode a spatial *degree of interest (DoI)* corresponding to the LoD value $l$. There are different lens types w.r.t. the mask reference space, e.g., scene lenses (world space), camera lenses (camera space), or viewport lenses (screen space).

## 5.2 Level-of-Detail Sampling and Mapping

The rendering and LoD mapping process, as outlined in Figure 11, consists of a geometry rasterization pass described in Section 5.2.1 followed by a deferred LoD mapping pass responsible for sampling the aggregated data structures (Section 5.2.2) and subsequent value quantization (Section 5.2.3), as well as the final color mapping.

*5.2.1 Geometry Rasterization.* To minimize the number of sampling operations, the visualization stage uses a *deferred shading* [45] approach. This uses a scene-rendering pass to rasterize the reference geometry into a set of G-Buffers storing various geometric attributes, thus creating

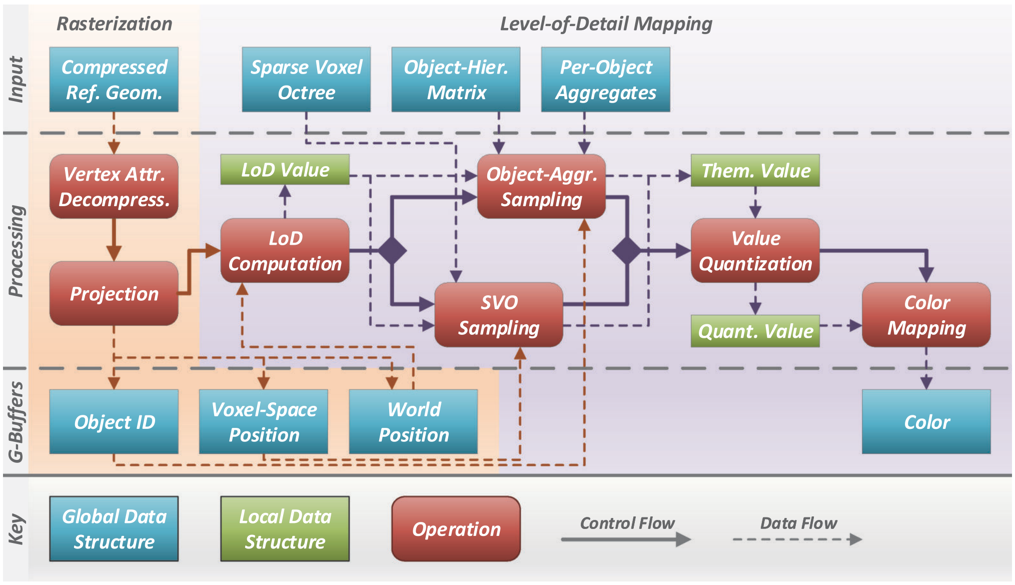


Fig. 11. Schematic overview of the LoD visualization process consisting of a geometry rasterization pass followed by a deferred LoD mapping pass.

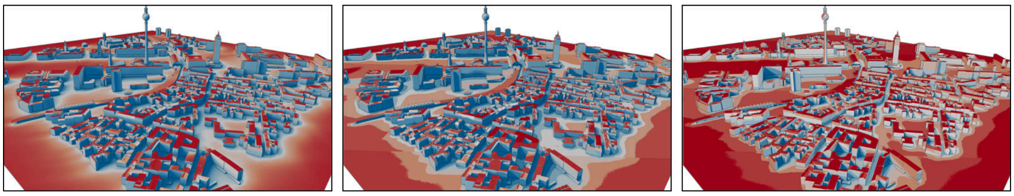
an image-based representation. Sampling and lighting operations are deferred to an image-based shading pass, and thus these are only performed for fragments visible in the output image. This improves rendering performance, as sampling operations are potentially complex, depending on the selected configuration, and include up to 16 SVO traversals per fragment, as described in Section 5.2.2. The implementation of the geometry rasterization pass is straightforward, as it only includes the projection of vertex positions from world-space coordinates to window-space coordinates.

5.2.2 *Level-of-Detail Sampling.* Subsequent to the geometry rasterization, the thematic detail levels are mapped onto the image-based scene representation using procedural texturing approaches with each aggregation type requiring a specific sampling method, as described in the following sections. In general, both sampling methods take the normalized LoD value $l$ determined using the interaction techniques described in Section 5.1 as input parameters.

*Sampling of Scene-Based Aggregation.* The results of scene-based aggregation are mapped onto the reference geometry using procedural texturing based on the values represented by the different SVO levels. Given a fragment position $\vec{p} \in \mathbb{R}^3$ in world-space coordinates, the per-pixel LoD value $l$, and an SVO with $n$ levels, sampling is performed as follows: first, $\vec{p}$ is projected to voxel-space coordinates as in Equation (1), using an orthogonal projection matrix $\mathbf{O}$ corresponding to the bounding cube used during scene voxelization, and resulting in the voxel coordinates $\vec{v}$. This step can be executed during vertex processing of the initial geometry pass and the results interpolated by the rasterizer:

$$\vec{v} := \lfloor (\mathbf{O} \cdot \vec{p} \cdot 0.5 + 0.5) \cdot 2^{n-1} \rfloor. \tag{1}$$

Next, the LoD value $l$ is mapped to an SVO level $k := \lceil l \cdot n \rceil$, and subsequently, the SVO is traversed down to the node with the coordinates $\vec{v}$ on the level $k$ and the associated thematic attribute value is fetched from the attribute texture. During sampling, different filtering techniques can be



(a) Original. (b) Linear quantization ($q_{max} = 8$, $p = 0.0$). (c) Non-linear quantization ($q_{max} = 8, p = 0.5$).

Fig. 12. Examples of different LoD quantization configurations with distance-based level of detail (dataset: Berlin).

applied to smooth transitions between neighboring voxels with different thematic attribute values by interpolating their values:

**Nearest** This is the simplest filter and does not do any interpolation.

**Trilinear** To avoid hard edges between neighboring voxels, the trilinear filter performs linear interpolation in all three spatial dimensions, requiring eight SVO traversals.

**Quadlinear** To avoid hard edges between different LoD levels, the quadlinear filter performs trilinear filtering in both LoD levels and linearly interpolates between the results, thus requiring 16 traversals in total.

*Sampling of Object-Based Aggregation.* To sample the results of object-based aggregation, the LoD value $l$ is scaled to the maximum depth of the scene's object hierarchy $d$ to obtain the required hierarchy level $h := \lceil l \cdot d \rceil$. Subsequently, given the leaf object identifier $o_l$, the object identifier $o_h$ of the corresponding object on level $h$ can be obtained from the two-dimensional object-hierarchy matrix using the coordinates $c := (h, o_l)$. Finally, $o_h$ is used to look up the aggregated thematic value in the per-object aggregate list. As per-object aggregates represent discrete scene objects, no spatial filtering techniques are required; however, linear filtering between different detail levels can be applied.

*5.2.3 Thematic Value Quantization.* After performing sampling to obtain the aggregated thematic data values, an additional *value quantization* can optionally be applied. The quantization, i.e., the discretization of the thematic attribute value range, is another approach to enable level-of-detail rendering for thematic data by mapping the LoD value $l$ to the number of quantization levels $q := \lceil l \cdot q_{max} \rceil$ based on a user-configurable $q_{max} \in \mathbb{N}^+$. The quantized thematic data value $v_q$ is computed as in Equation (2), which is based on the work of Lux et al. [35] and supports nonlinear mapping through the user-configurable power level $p \in [-1, 1] \subset \mathbb{R}$:

$$v_q := \frac{\lfloor v^p \cdot q \rfloor}{q} + \frac{1}{2q}. \tag{2}$$

Activated data quantization provides clarity in the presentation of scene-based aggregation results and further reduces visual clutter [52]. Using this approach, regions with higher detail levels are visualized with fine gradations, while others with less detail only use a few distinct values, as illustrated in Figure 12.

## 6 DISCUSSION

To confirm the feasibility of our approach for real-time level-of-detail generation and visualization of time-variant thematic data, we conducted experiments using three virtual city models of

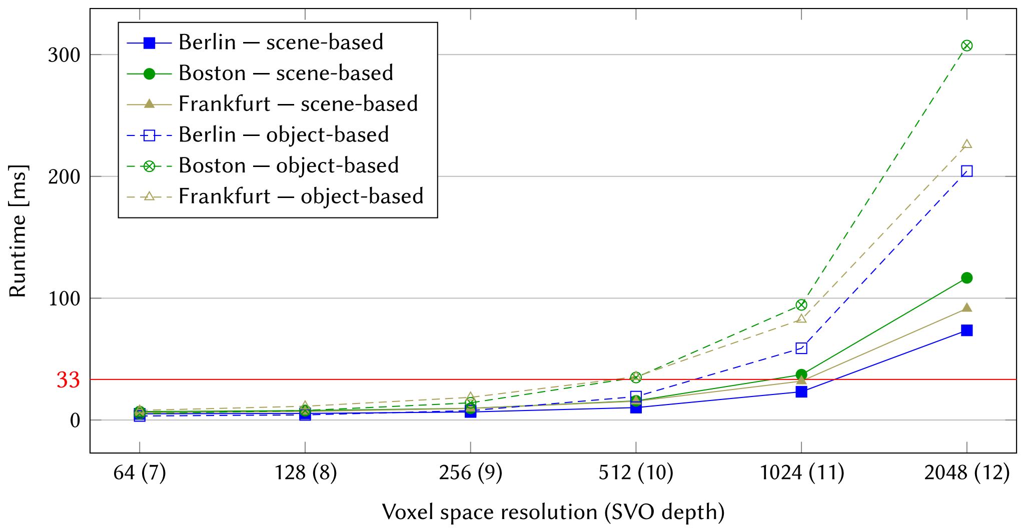


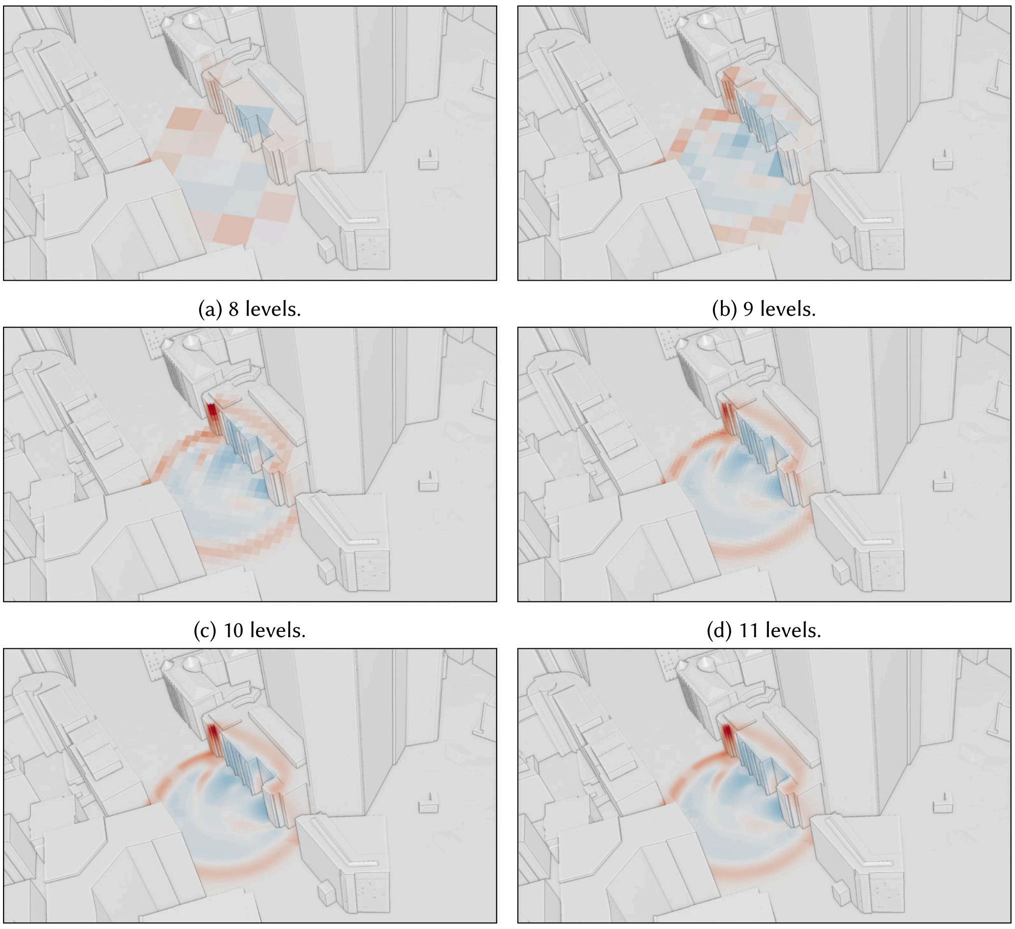
Fig. 13. Comparison of total runtimes in milliseconds for object-based and scene-based aggregation techniques w.r.t. different voxel space resolutions and corresponding sparse voxel octree depth.

varying geometric complexity, ranging from 270,000 to 840,000 vertices. Two of those models included static solar potential analyses encoded as 2D surface textures, while the third included the results of a pressure wave propagation simulation encoded as volumetric raster data with 201 samples. We then measured the runtimes of the LoD generation and visualization stages using a consumer-grade GPU (*NVIDIA GeForce GTX 970* with 4 GB VRAM) and varying quality settings, i.e., different resolutions of the initial voxelization step, as well as filtering methods during rendering. For the purpose of this work, a frame rate of 30 frames per second (FPS) is considered real time, corresponding to a time-to-frame of 33.3 ms, and hence this constitutes the target runtime for real-time operations.

Figure 13 summarizes the results, showing that scene-based aggregation can be performed in real time for all three datasets up to a voxel space resolution of $512^3$, corresponding to 10 SVO levels. Object-based aggregation is generally more time-consuming, which is due to the high synchronization overhead of merging voxel fragments per object: e.g., in case of the Boston model with a voxel space resolution of $2040^3$, on average 7,278 voxel fragments contribute to the aggregation result of a single object. For detailed results, see Section A in the online supplementary material.

Maximizing the voxelization resolution is required to preserve fine details during scene-based aggregation, corresponding to a deeper sparse voxel octree structure with more detailed levels. This requirement is less prominent for object-based aggregation, as the maximum detail level of the aggregated data is limited by the granularity of the finest object hierarchy level, which is typically not that high (e.g., walls in a virtual city model).

Figure 14 demonstrates this runtime versus quality tradeoff for scene-based aggregation by example of the pressure wave dataset for sparse voxel octrees with eight to 12 levels. For 12 levels (Figure 14(e)), basically no difference to the original thematic data (Figure 14(f)) is visible, while eight and nine levels (Figure 14(a) and Figure 14(b)) are not suitable for detailed analyses. Starting from an SVO with 10 levels, most details are preserved, making this the minimum requirement for thematic data analysis. This matches the upper bound for the real-time execution described above and therefore confirms that our technique can generate required LoD with sufficient detail in real time.



(a) 8 levels. (b) 9 levels.

(c) 10 levels. (d) 11 levels.

(e) 12 levels. (f) Original thematic data.

Fig. 14. Comparison of resulting visual quality for scene-based aggregation with different numbers of SVO levels (dataset: Frankfurt).

For dynamic datasets, 10 levels is the maximum achievable detail level on the test hardware without breaking the real-time constraint; however, 11 levels seem feasible for the next hardware generation. Although the memory consumption is the limiting factor in case of static datasets, the test hardware provides sufficient memory for 11 or 12 SVO levels (ca. 0.3 GB and 1.1 GB, respectively), thus preserving the finest details. As memory consumption and runtime are both tightly coupled to the number of voxel fragments and nodes, there is no space versus runtime tradeoff to consider.

The LoD generation techniques presented in this article can be applied to both static and dynamic thematic data visualization. Figure 15 demonstrates the scene-based aggregation for time-variant thematic data (pressure wave propagation). The thematic data consist of 200 frames (i.e., discrete representations in time) of 3D raster data that is spatially referenced w.r.t. a virtual 3D city model. Each cell stores the simulated pressure at its respective location and point in time. Using our level-of-detail generation technique, we can generate required detail levels on the fly during visualization to support comprehensive visual analysis in real time.

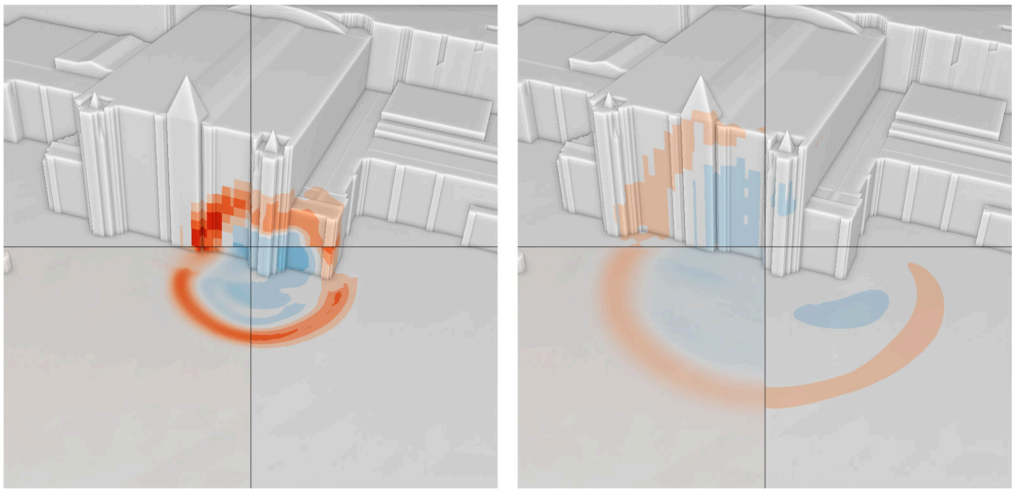


Fig. 15. Two frames of summarizing level-of-detail visualization variants of pressure waves (data clockwise: original thematic data, aggregated data, aggregated and quantized data, quantized data only; dataset: Frankfurt).

Throughout the article, the aggregation techniques were demonstrated using results of solar potential simulation runs for different virtual 3D city models (Berlin, Boston, Chemnitz). This type of thematic data is encoded using 2D textures associated with geometric primitives of respective 3D building objects. Due to the static nature of this data, an on-the-fly level-of-detail generation does not seem necessary; however, with real-time LoD, the user can change aggregation functions and parameters on demand and instantly view the result, thus increasing the interactivity of the visualization and avoiding the user losing focus [36].

The currently supported aggregation functions include `minimum` and `maximum` applicable to ordinal, interval, and ratio-scaled data, as well as `average` for interval and ratio-scaled data; however, common central tendencies for ordinal and nominal data are `median` and `mode`, which are not trivially computable in the hierarchical fashion of a sparse voxel octree. In future research, we plan to implement an alternative aggregation scheme that computes a value histogram per voxel, which then can be hierarchically aggregated using bin-wise summation. Subsequently, approximate values for `minimum`, `maximum`, `average`, `median`, and `mode` are derived from the aggregated histogram. For ordinal and nominal data, choosing the number of histogram bins equal to the number of data classes instead allows for the exact computation of these functions.

In addition, we plan to amend our aggregation techniques with techniques for the detection and visualization of outliers, i.e., data values that differ significantly from their surrounding area and that are typically considered an important aspect of the dataset [6]. Using the described spatial aggregation approaches, outliers may be blended into their surroundings on levels with low resolution and thus hidden from the viewer. Temporal aggregation of time-variant thematic data presents an interesting research opportunity as well and can be implemented on top of the presented aggregation approaches.

## APPENDIX

## A SUPPLEMENTARY MATERIAL

See the supplementary materials in the online version.

## ACKNOWLEDGMENTS

The author would like to thank the anonymous reviewers for their valuable comments that improved the article. This work was funded by the German Federal Ministry of Education and Research (BMBF) in the InnoProfile-Transfer research group “4DnDVis” under grantnumber 03IPT516Y.

Data courtesy and copyright of *Berlin Partner für Wirtschaft und Technologie GmbH* (Berlin city model) and *Stadtvermessungsamt Frankfurt a. M.* (Frankfurt city model).

## REFERENCES

[1] Natalia V. Andrienko and Gennady L. Andrienko. 2011. Spatial generalization and aggregation of massive movement data. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 17, 2 (2011), 205–219. Retrieved from http://dx.doi.org/10.1109/TVCG.2010.44.

[2] Nathan Andrysco and Xavier Tricoche. 2010. Matrix trees. *Computer Graphics Forum* 29, 3 (2010), 963–972. Retrieved from http://dx.doi.org/10.1111/j.1467-8659.2009.01709.x.

[3] James F. Blinn. 1977. Models of light reflection for computer synthesized pictures. In *Proceedings of the 4th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH’77)*. ACM, New York, 192–198. Retrieved from http://dx.doi.org/10.1145/563858.563893.

[4] Alexander Bock, Alexander Kleiner, Jonas Lundberg, and Timo Ropinski. 2014. Supporting urban search & rescue mission planning through visualization-based analysis. In *Proceedings of the 19th International Workshop on Vision, Modeling and Visualization (VMV’14)*. Eurographics Association, 47–54. Retrieved from http://dx.doi.org/10.2312/vmv.20141275.

[5] Henrik Buchholz and Jürgen Döllner. 2005. View-dependent rendering of multiresolution texture-atlases. In *Proceedings of the IEEE Visualization 2005 (VIS’05)*. 215–222. Retrieved from http://dx.doi.org/10.1109/VISUAL.2005.1532798.

[6] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)* 41, 3 (2009), 15.

[7] Andy Cockburn, Amy Karlson, and Benjamin B. Bederson. 2009. A review of overview + detail, zooming, and focus + context interfaces. *ACM Computing Surveys (CSUR)* 41, 1, Article 2 (2009), 31 pages. Retrieved from http://dx.doi.org/10.1145/1456650.1456652.

[8] Jonathan Cohen, Marc Olano, and Dinesh Manocha. 1998. Appearance-preserving simplification. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH’98)*. ACM, New York, 115–122. Retrieved from http://dx.doi.org/10.1145/280814.280832.

[9] Cyril Crassin and Simon Green. 2012. Octree-based sparse voxelization using the GPU hardware rasterizer. In *OpenGL Insights*, Patrick Cozzi and Christophe Riccio (Eds.). CRC Press, 303–319.

[10] Christopher DeCoro and Natalya Tatarchuk. 2007. Real-time mesh simplification using the GPU. In *Proceedings of the 2007 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D’07)*. ACM, New York, 161–166. Retrieved from http://dx.doi.org/10.1145/1230100.1230128.

[11] Ioannis Delikostidis, Juri Engel, Bas Retsios, Corné P. J. M. van Elzakker, Menno-Jan Kraak, and Jürgen Döllner. 2013. Increasing the usability of pedestrian navigation interfaces by means of landmark visibility analysis. *Journal of Navigation* 66, 4 (2013), 523–537. Retrieved from http://dx.doi.org/10.1017/S0373463313000209.

[12] Jürgen Döllner and Henrik Buchholz. 2005. Continuous level-of-detail modeling of buildings in 3D city models. In *Proceedings of the 13th Annual ACM International Workshop on Geographic Information Systems (GIS’05)*. ACM, New York, 173–181. Retrieved from http://dx.doi.org/10.1145/1097064.1097089.

[13] Zhao Dong, Wei Chen, Hujun Bao, Hongxin Zhang, and Qunsheng Peng. 2004. Real-time voxelization for complex polygonal models. In *Proceedings of the 12th Pacific Conference on Computer Graphics and Applications (PG’04)*. IEEE Computer Society, 43–50. Retrieved from http://dl.acm.org/citation.cfm?id=1025128.1026026.

[14] Nobert Edsall, Gennady Andrienko, Natalia Andrienko, and Barbara Buttenfield. 2008. *Manual of Geographic Information Systems*. American Society for Photogrammetry and Remote Sensing (ASPRS).

[15] Elmar Eisemann and Xavier Décoret. 2006. Fast scene voxelization and applications. In *Proceedings of the 2006 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D’06)*. ACM SIGGRAPH, 71–78. Retrieved from http://maverick.inria.fr/Publications/2006/ED06.

[16] Niklas Elmqvist and Jean-Daniel Fekete. 2010. Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 16, 3 (2010), 439–454. Retrieved from http://dx.doi.org/10.1109/TVCG.2009.84.

[17] Juri Engel, Amir Semmo, Matthias Trapp, and Jürgen Döllner. 2013. Evaluating the perceptual impact of rendering techniques on thematic color mappings in 3D virtual environments. In *Proceedings of 18th International Workshop on*

*Vision, Modeling and Visualization (VMV'13)*, Michael Bronstein, Jean Favre, and Kai Hormann (Eds.). Eurographics Association, 25–32.

[18] Billur Engin, Burcin Bozkaya, and Selim Balcisoy. 2009. Introducing level of detail to 3D thematic maps. In *GeoViz Hamburg 2009*. http://geovisualisierung.net/geoviz_hamburg/papers/08_2_Engin.pdf.

[19] Tassilo Glander and Jürgen Döllner. 2007. Cell-based generalization of 3D building groups with outlier management. In *Proceedings of the 15th Annual ACM International Symposium on Advances in Geographic Information Systems (GIS'07)*. ACM, New York, Article 54, 4 pages. Retrieved from http://dx.doi.org/10.1145/1341012.1341078.

[20] Li Guangming, Tian Jie, Zhao Mingchang, He Huiguang, and Zhang Xiaopeng. 2002. A new mesh simplification algorithm combining half-edge data structure with modified quadric error metric. In *Object Recognition Supported by User Interaction for Service Robots*, Vol. 2. 659–662. Retrieved from http://dx.doi.org/10.1109/ICPR.2002.1048388.

[21] Dongliang Guo, Junlan Nie, Meng Liang, Yu Wang, Yanfen Wang, and Zhengping Hu. 2015. View-dependent level-of-detail abstraction for interactive atomistic visualization of biological structures. *Computers & Graphics* 52 (2015), 62–71. Retrieved from http://dx.doi.org/10.1016/j.cag.2015.06.008.

[22] Mark A. Harrower and Cynthia A. Brewer. 2003. ColorBrewer.org: An online tool for selecting color schemes for maps. *Cartographic Journal* 40 (2003), 27–37. Retrieved from http://ColorBrewer.org/.

[23] Jan-Henrik Haunert. 2008. *Aggregation in Map Generalization by Combinatorial Optimization.* Ph.D. Dissertation.

[24] Hugues Hoppe. 1999. New quadric metric for simplifying meshes with appearance attributes. In *Proceedings of the IEEE Visualization'99 (VISUALIZATION'99)*. IEEE Computer Society. Retrieved from http://dl.acm.org/citation.cfm?id=832273.834119.

[25] Ladislav Hrabcak and Arnaud Masserann. 2012. Asynchronous buffer transfers. In *OpenGL Insights*, Patrick Cozzi and Christophe Riccio (Eds.). CRC Press, 391–414.

[26] Marta Indulska and Maria Elzbieta Orlowska. 2002. On aggregation issues in spatial data management. *Australian Computer Science Communications* 24, 2 (2002), 75–84. Retrieved from http://dx.doi.org/10.1145/563932.563915.

[27] Viktor Kämpe, Erik Sintorn, and Ulf Assarsson. 2013. High resolution sparse voxel DAGs. *ACM Transactions on Graphics (TOG)* 32, 4, Article 101 (2013), 13 pages. Retrieved from http://dx.doi.org/10.1145/2461912.2462024.

[28] Thomas H. Kolbe, Gerhard Gröger, and Lutz Plümer. 2005. CityGML: Interoperable access to 3D city models. In *Geo-Information for Disaster Management*, Peter van Oosterom, Siyka Zlatanova, and Elfriede Fendel (Eds.). Springer, Berlin, 883–899. Retrieved from http://dx.doi.org/10.1007/3-540-27468-5_63.

[29] Matthias Labschütz, Stefan Bruckner, Eduard Gröller, Markus Hadwiger, and Peter Rautek. 2016. JiTTree: A just-in-time compiled sparse GPU volume data structure. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 22, 1 (2016), 1025–1034. Retrieved from http://dx.doi.org/10.1109/TVCG.2015.2467331.

[30] Samuli Laine and Tero Karras. 2011. Efficient sparse voxel octrees. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 17 (2011), 1048–1059. Retrieved from http://dx.doi.org/10.1109/TVCG.2010.240.

[31] Eugene Lapidous and Guofang Jiao. 1999. Optimal depth buffer for low-cost graphics hardware. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware (HWWS'99)*. ACM, New York, 67–73. Retrieved from http://dx.doi.org/10.1145/311534.311579.

[32] Thomas Liebich, Yoshinobu Adachi, James Forester, Juha Hyvarinen, Kari Karstila, and Jeffrey Wix. 2006. *Industry Foundation Classes IFC2×3. International Alliance for Interoperability*, 467–476.

[33] Haik Lorenz and Jürgen Döllner. 2010. 3D feature surface properties and their application in geovisualization. *Computers, Environment and Urban Systems (CEUS)* 34, 6 (2010), 476–483.

[34] David Luebke, Benjamin Watson, Jonathan D. Cohen, Martin Reddy, and Amitabh Varshney. 2002. *Level of Detail for 3D Graphics.* Elsevier Science Inc.

[35] Roland Lux, Matthias Trapp, Amir Semmo, and Jürgen Döllner. 2013. Interactive projective texturing for non-photorealistic shading of technical 3D models. In *Proceedings of 11th Theory and Practice of Computer Graphics 2013 Conference (TPCG'13)*, Silvester Czanner and Wen Tang (Eds.). 101–108.

[36] Jakob Nielsen. 1993. *Usability Engineering*. Morgan Kaufmann.

[37] Alexandros Papageorgiou and Nikos Platis. 2015. Triangular mesh simplification on the GPU. *Visual Computer* 31, 2 (2015), 235–244. Retrieved from http://dx.doi.org/10.1007/s00371-014-1039-x.

[38] Julius Parulek, Daniel Jönsson, Timo Ropinski, Stefan Bruckner, Anders Ynnerman, and Ivan Viola. 2014. Continuous levels-of-detail and visual abstraction for seamless molecular visualization. *Computer Graphics Forum* 33, 6 (2014), 276–287. Retrieved from http://dx.doi.org/10.1111/cgf.12349.

[39] Mark Pauly, Markus Gross, and Leif P. Kobbelt. 2002. Efficient simplification of point-sampled surfaces. In *Proceedings of the IEEE Visualization, 2002 (VIS'02)*. 163–170. Retrieved from http://dx.doi.org/10.1109/VISUAL.2002.1183771.

[40] Mark Segal and Kurt Akeley. 2015. *The OpenGL® Graphics System: A Specification (Version 4.5 (Core Profile) - May 28, 2015)*. The Khronos Group Inc.

[41] Amir Semmo, Matthias Trapp, Jan Eric Kyprianidis, and Jürgen Döllner. 2012. Interactive visualization of generalized virtual 3D city models using level-of-abstraction transitions. *Computer Graphics Forum* 31, 3 (2012), 885–894.

[42] Ben Shneiderman. 1996. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the 1996 IEEE Symposium on Visual Languages*. IEEE, 336–343.

[43] Georgiy Shurkhovetskyy, Natalia V. Andrienko, Gennady L. Andrienko, and Georg Fuchs. 2017. Data abstraction for visualizing large time series. *Computer Graphics Forum* 37, 1 (2017), 125–144. Retrieved from http://dx.doi.org/10.1111/cgf.13237.

[44] John Snow. 1855. *On the Mode of Communication of Cholera*. John Churchill. Retrieved from https://books.google.de/books?id=-N0_AAAAcAAJ.

[45] Brice Tebbs, Ulrich Neumann, John Eyles, Greg Turk, and David Ellsworth. 1989. *Parallel Architectures and Algorithms for Real-Time Synthesis of High Quality Images Using Deferred Shading*. Technical Report. DTIC Document.

[46] The Khronos Group Inc. 2016. *The OpenGL® Shading Language*.

[47] Matthias Thöny, Markus Billeter, and Renato Pajarola. 2015. Vision paper: The future of scientific terrain visualization. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS'15)*. ACM, New York, Article 13, 4 pages. Retrieved from http://dx.doi.org/10.1145/2820783.2820884.

[48] Christian Tominski, Stefan Gladisch, Ulrike Kister, Raimund Dachselt, and Heidrun Schumann. 2014. A survey on interactive lenses in visualization. In *EuroVis State-of-the-Art Reports*, R. Borgo, R. Maciejewski, and I. Viola (Eds.). Eurographics Association. Retrieved from http://dx.doi.org/10.2312/eurovisstar.20141172.

[49] Matthias Trapp. 2013. *Interactive Rendering Techniques for Focus+Context Visualization of 3D Geovirtual Environments*. Ph.D. Dissertation.

[50] Matthias Trapp and Jürgen Döllner. 2009. Dynamic mapping of raster-data for 3D geovirtual environments. In *2009 13th International Conference Information Visualization (IV'09)*. 387–392. Retrieved from http://dx.doi.org/10.1109/IV.2009.28.

[51] Matthias Trapp, Tassilo Glander, Henrik Buchholz, and Jürgen Döllner. 2008. 3D generalization lenses for interactive focus + context visualization of virtual city models. In *Proceedings of the 12th International Conference Information Visualisation (IV'08)*. IEEE Computer Society Press, 356–361.

[52] Jun Yan and Jean-Claude Thill. 2008. Visual exploration of spatial interaction data with self-organizing maps. In *Self-Organising Maps*. John Wiley & Sons, 67–85. Retrieved from http://dx.doi.org/10.1002/9780470021699.ch4.

[53] Fan Zhang, Vincent Tourre, and Guillaume Moreau. 2013. A general strategy for semantic levels of detail visualization in urban environment. In *Eurographics Workshop on Urban Data Modelling and Visualisation*, Vincent Tourre and Gonzalo Besuievsky (Eds.). Eurographics Association, 33–36. Retrieved from http://dx.doi.org/10.2312/UDMV/UDMV13/033-036.

[54] Jianting Zhang, Simin You, and Le Gruenwald. 2012. High-performance online spatial and temporal aggregations on multi-core CPUs and many-core GPUs. In *Proceedings of the 15th International Workshop on Data Warehousing and OLAP (DOLAP'12)*. ACM, New York, 89–96. Retrieved from http://dx.doi.org/10.1145/2390045.2390060.

[55] Michael Zinsmaier, Ulrik Brandes, Oliver Deussen, and Hendrik Strobelt. 2012. Interactive level-of-detail rendering of large graphs. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 18, 12 (2012), 2486–2495. Retrieved from http://dx.doi.org/10.1109/TVCG.2012.238.

Received October 2017; revised June 2018; accepted June 2018